A Double-Space and Double-Norm Ensembled Latent Factor Model for Highly Accurate Web Service QoS Prediction

Di Wu, *Member, IEEE*, Peng Zhang, *Student Member, IEEE*, Yi He, *Member, IEEE*, and Xin Luo, *Senior Member, IEEE*

Abstract—Quality-of-Service (QoS), which describes the non-functional characteristics of Web service, is of great significance in service selection. Since users cannot invoke all services to obtain the corresponding QoS data, QoS prediction becomes a hot yet thorny issue. To date, a latent factor analysis (LFA)-based QoS predictor is one of the most successful and popular approaches to address this issue. However, current LFA-based QoS predictors are mostly modeled on inner product space with an L_2-norm-oriented *Loss* function only. They cannot comprehensively represent the characteristics of target QoS data to make accurate predictions because inner product space and L_2-norm have their respective limitations. To address this issue, this study proposes a Double-space and Double-norm Ensembled Latent Factor (D^2E-LF) model. Its main idea is three-fold: 1) Double-space—inner product space and distance space are employed to model two kinds of LFA-based QoS predictors, respectively, 2) Double-norm—both of these two predictors adopt an L_1-and-L_2-norm-oriented *Loss* function, and 3) Ensembled—building an ensemble of these two predictors by a weighting strategy. By doing so, D^2E-LF integrates multi-merits originating from inner product space, distance space, L_1-norm, and L_2-norm, making it achieve highly accurate QoS prediction. Experiments on two real-world QoS datasets demonstrate that D^2E-LF has significantly higher prediction accuracy than state-of-the-art models.

Index Terms—Web service, service selection, Quality-of-Service (QoS), latent factor analysis, missing data prediction, big data

1. Introduction

In This Era of Cloud Computing, Web service is a fundamental and essential component of the service-oriented architecture of industrial applications [1], [2], [3]. It is an interface to exchange information between different computing terminals [4]. In general, many services can provide similar functionality to users [5]. Hence, selecting an appropriate service for a target user from numerous candidates becomes the primary concern in practical applications [6].

Except for functionality, non-functional characteristics of Web service are also crucial. They represent the quality of services to ensure the reliability of the ultimate applications [6], [7]. The non-functional characteristics are collectively called quality of service (QoS) [1], [2], [3], including response time, invocation failure rate, throughput, capacity, robustness, availability, etc. The optimal service can be easily selected from many potential ones with reliable QoS data. Generally, QoS data can be obtained through a warm-up test [8], [9]. However, invocations of business services are usually not free. Besides, it is time-consuming to evaluate all the candidates. As a result, the warm-up test is impractical. Under such situation, QoS prediction plays an essential role in obtaining QoS data [10], [11].

Commonly, since a user cannot touch all of the available services in real-world scenarios, the original data of QoS prediction is a sparse QoS matrix [8], [9], where each row denotes a user, each column denotes a service, and each observed entry denotes an observed historical record that is a user's invocation on a service. The principle of QoS prediction is to predict the missing data of a sparse QoS matrix based on its observed ones. Among numerous QoS prediction approaches, a latent factor analysis (LFA)-based one is widely studied and adopted owing to its high efficiency and scalability [12], [13], [14], [15], [16].

Given a sparse QoS matrix, an LFA-based predictor aims to train two latent factor matrices to make a low-rank approximation by minimizing the differences between observed data and predictions [8], [9]. Although various proposed LFA-based predictors are different in model design, they have a common characteristic—the aforementioned differences are minimized on inner product space with an L_2-norm-oriented

- *Di Wu and Xin Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China, and also with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China. E-mail: {wudi.cigit, luoxin21}@gmail.com.*
- *Peng Zhang is with the School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: s190231042@stu.cqupt.edu.cn.*
- *Yi He is with the Old Dominion University, Norfolk, VA 23529 USA. E-mail: yihe@cs.odu.edu.*

Manuscript received 18 Dec. 2021; revised 29 Apr. 2022; accepted 23 May 2022. Date of publication 27 May 2022; date of current version 10 Apr. 2023.
This work was supported in part by the National Natural Science Foundation of China under grants 62176070, 62002337, and 61902370, in part by the Natural Science Foundation of Chongqing (China) under grants cstc2019jcyj-msxmX0578, in part by the CAS "Light of West China" Program, in part by the Guangdong Province Universities and College Pearl River Scholar Funded Scheme (2019), and in part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008).
(Corresponding author: Xin Luo.)
Digital Object Identifier no. 10.1109/TSC.2022.3178543

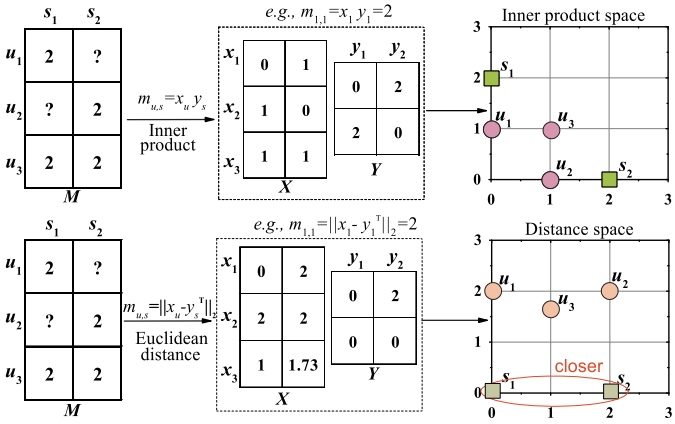


Fig. 1. An example: the distributions of latent factors on inner product space and Euclidean distance space in representing a sparse QoS matrix.

Loss [8], [9], [17]. However, since inner product space and L_2-norm have their respective limitations [17], [18], [19], the LFA-based predictors originating from them cannot comprehensively represent the target QoS matrix's characteristics to make accurate predictions. Next, we respectively illustrate their limitations.

First is inner product space. Although it can capture the global similarity among users and services, it easily ignores their fine-grained similarity, i.e., the local similarity among users or services [18], [19]. On the contrary, distance space can easily capture such local similarity [18], [19]. Example 1 illustrates the difference between inner product space and distance space in Fig 1.

Example 1. Given a sparse QoS matrix M with three users u_1, u_2, and u_3, and two services s_1 and s_2, the invocation between a user u and a service s is represented as $m_{u,s}$, e.g., $m_{1,1} = 2$. From M, we see that both u_1 and u_2 are similar to u_3 as $m_{1,1} = m_{3,1} = 2$ and $m_{2,2} = m_{3,2} = 2$, which indicates that u_1, u_2, and u_3 are all similar. The similarity among u_1, u_2, and u_3 can be called 'global similarity' because it is discovered from all the known invocations of M. Besides, it is easy to deduce that s_1 and s_2 are similar as $m_{3,1} = m_{3,2} = 2$. The similarity between s_1 and s_2 can be called 'local similarity' because it is discovered from u_3's invocations only. To represent M's invocations, inner product space adopts the inner product of two latent factor vectors—$m_{u,s} = x_u y_s$, while distance space adopts the Euclidean distance of them—$m_{u,s} = ||x_u - y_s^{\mathrm{T}}||_2$, where x_u is a row vector of X corresponding to a user u, y_s is a column vector of Y corresponding to a service s, and X and Y are user and service latent factor matrices, respectively. The representation results are visualized on the two-dimensional distribution maps, where the global similarity among u_1, u_2, and u_3 is well represented because u_1, u_2, and u_3 are close to each other. However, the local similarity between s_1 and s_2 is not well represented as s_1 and s_2 are far away from each other. While in distance space, the local similarity is better represented as s_1 and s_2 are closer.

The second is L_2-norm. As well known, L_2-norm is smooth when the predicted results and ground truths are close to each other [17]. Hence, a predictor with an L_2-norm-oriented *Loss* function can usually obtain stable predictions. However, as L_2-norm is sensitive to outliers, such a predictor's robustness cannot be guaranteed when a QoS matrix is mixed with noise data. On the contrary, since L_1-norm is not as sensitive as L_2-norm to outliers [17], [20], we expect that its robustness can be greatly improved by incorporating L_1-norm into a predictor's *Loss* function. Example 2 illustrates the difference between L_1-norm and L_2-norm in Fig. 2.

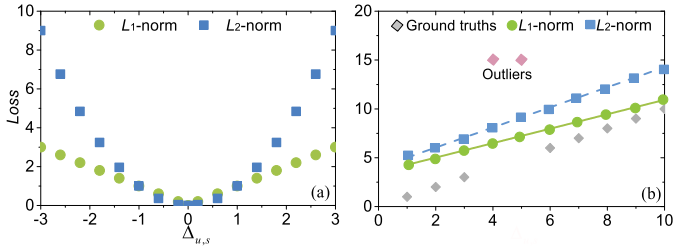


Fig. 2. An example: The difference between L_1-norm and L_2-norm in formulating the Loss function of an LFA-based QoS Predictor [17].

Example 2. In an LFA-based QoS predictor, its *Loss* function measures the differences between ground truths and predictions. Let $\Delta_{u,s}$ denote the difference between ground truth $m_{u,s}$ and its prediction $\hat{m}_{u,s}$. An L_1-norm- and L_2-norm-oriented *Losses* are $Loss = |\Delta_{u,s}|$ and $Loss = (\Delta_{u,s})^2$, respectively. Their functional relationships are shown in Fig. 2a, where we see that L_1-norm is less sensitive to $\Delta_{u,s}$ than L_2-norm. Therefore, L_1-norm- oriented *Loss* is more robust to outliers than L_2-norm in regressing the ground truth QoS data, as shown in Fig. 2b.

We conclude from the above two examples that it is not the best choice to develop an LFA-based QoS predictor based on inner product space with an L_2-norm-oriented *Loss* function only. To enjoy the multi-merits originating from inner product space, distance space, L_1-norm, and L_2-norm all at once, this paper proposes a Double-space and Double-norm Ensembled Latent Factor (D^2E-LF) model. Its main idea is three-fold: 1) Double-space—inner product space and distance space are employed to model two kinds of LFA-based QoS predictors, respectively, 2) Double-norm—both of these two predictors adopt an L_1-and-L_2-norm-oriented *Loss* function, and 3) Ensembled—building an ensemble of these two predictors by a weighting strategy. By doing so, D^2E-LF achieves highly accurate QoS prediction and significantly outperforms its peers in terms of prediction accuracy. The main contributions include:

- A D^2E-LF model is proposed. It can comprehensively represent a sparse QoS matrix's characteristics to accurately predict its missing data;
- Algorithm design and analysis are given for the proposed D^2E-LF model;
- Detailed experiments on two real-world QoS datasets are performed to evaluate the proposed D^2E-LF, including comparison with eight state-of-the-art models and analyses of its characteristics.

Section 2 states the preliminaries. Section 3 presents the D^2E-LF model. Section 4 reveals experimental results. Section 5 concludes this paper.

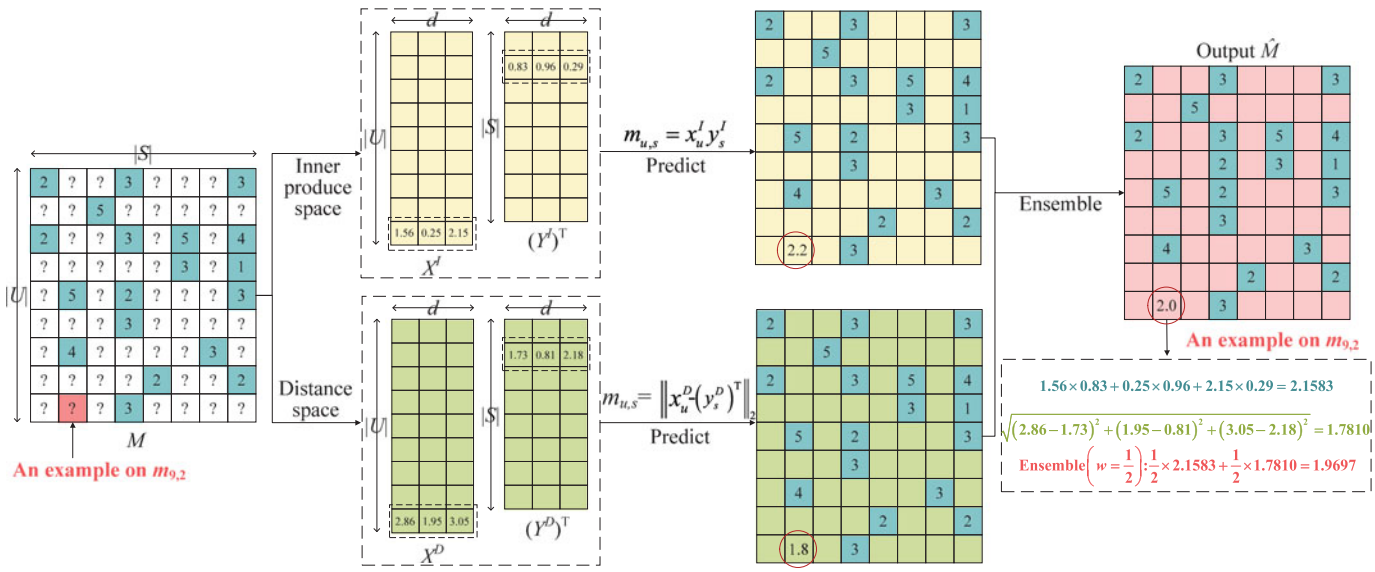


Fig. 3. The architecture of the proposed D^{2}E-LF model.

2 Preliminaries

Definition 1 Problem of a QoS predictor. *Given a user set* U *and a service set* S*, then we can get a* $|\mathrm{U}|\times|\mathrm{S}|$ *matrix* M *that each element* $\mathrm{m}_{u,s}$ *denotes a historical record (e.g., response time) of user* u(u∈U) *on service* s(s∈S). *Let* $\mathrm{M_K}$ *and* $\mathrm{M_U}$ *denote* M*'s known and unknown entry sets, respectively. The problem of a QoS predictor is to predict each unknown element of* $\mathrm{M_U}$ *based on* $\mathrm{M_K}$ *accurately and efficiently.*

Definition 2 An LFA-based QoS predictor. *Given a QoS matrix* M*, an LFA-based QoS predictor is to train two latent factor matrices* X *of size* $|\mathrm{U}|\times\mathrm{d}$ *and* Y *of size* $\mathrm{d}\times|\mathrm{S}|$ *to make* M*'s rank-*d *approximation* $\hat{\mathrm{M}}$ *based on* $\mathrm{M_K}$ *only with* $\mathrm{d}\ll min\{|\mathrm{U}|, |\mathrm{S}|\}$*, where a row vector of* X *reflect a user's characteristics and a column vector of* Y *reflects a service's characteristics in the latent factor space, respectively.*

To model an LFA-based QoS predictor, it is essential to design a *Loss* function to measure the differences between M and $\hat{M}$ [10], [11], [21]. Currently, most of existing predictors adopt the inner product space with an L_2-norm to do that as follows.

$$\arg\min \varepsilon(X,Y)=\frac{1}{2}\left\|\Omega\odot\left(M-\hat{M}\right)\right\|_{L_2}^2=\frac{1}{2}\|\Omega\odot(M-XY)\|_{L_2}^2, \tag{1}$$

where $\odot$ denotes the Hadamard product and Ω is binary index matrix whose entry at u-th row and s-th column is 1 if $m_{u,s}$ is observed and is 0 if $m_{u,s}$ is not observed. According to [10], [11], [21], regularization is crucial for avoiding overfitting. By incorporating Tikhonov regularization into (1), we have:

$$\arg\min \varepsilon(X,Y)=\frac{1}{2}\|\Omega\odot(M-XY)\|_{L_2}^2+\frac{\lambda}{2}\left(\|X\|_{L_2}^2+\|Y\|_{L_2}^2\right), \tag{2}$$

where λ is a hyperparameter that controls the regularization penalty intensity. Notably, since M is usually sparse, it is necessary to expand (2) into a density-oriented form to improve efficiency as follows [8], [9]:

$$\arg\min \varepsilon(X,Y)=\frac{1}{2}\sum_{m_{u,s}\in M_K}\left(m_{u,s}-\sum_{k=1}^{d}x_{u,k}y_{k,s}\right)^2+\frac{\lambda}{2}\sum_{m_{u,s}\in M_K}\left(\sum_{k=1}^{d}x_{u,k}^2+\sum_{k=1}^{d}y_{k,s}^2\right), \tag{3}$$

where $x_{u,k}$ denotes the u-th row and k-th column entry of X, and $y_{k,s}$ denotes the k-th row and s-th column entry of Y. Generally, (3) is minimized by stochastic gradient descent (SGD) [8], [9] to obtain the updating rules of $x_{u,k}$ and $y_{k,s}$ as follows:

$$for\ m_{u,s}\in M_K, \forall k\in\{1,2,\ldots,d\}: \begin{cases} x_{u,k}\leftarrow x_{u,k}-\eta\frac{\partial\varepsilon_{u,s}(X,Y)}{\partial x_{u,k}} \\ y_{k,s}\leftarrow y_{k,s}-\eta\frac{\partial\varepsilon_{u,s}(X,Y)}{\partial y_{k,s}}. \end{cases}, \tag{4}$$

where η is the learning rate and $\varepsilon_{u,s}(X,Y)$ is the instant state of (3) on a single invocation $m_{u,s}$.

3 The Proposed D2E-LF Model

Motivated by the two examples shown in Fig. 1 and 2, we design D^{2}E-LF to integrate the multi-merits originating from inner product space, distance space, L_1-norm, and L_2-norm. Fig. 3 depicts the architecture of D^{2}E-LF, which can be divided into three steps:

1) Inputing a QoS matrix M to train two kinds of predictors on
2) inner product space (yellow color) and distance space (green color) based on M_K of M, respectively.
3) Employing the two trained predictors to predict the missing data (marked '?') of M, respectively.
4) Ensembling the predictions of the two predictors by a weighting strategy to obtain the final output $\hat{M}$.

To illustrate the principle of D^{2}E-LF, an example of predicting $m_{9,2}$ is also given in Fig. 3. The predicted values of the two predictors are different, i.e., the predictions on inner produce space and distance space are 2.1583 and 1.7810, respectively. Finally, the two predictions are weighted to obtain the final prediction of 1.9697. Next, we describe D^{2}E-LF in detail.

3.1 Predictor on Inner Product Space (D2E-LF-1)

As analyzed in [17], [20], L_1-norm can be incorporated into an LFA model to improve its robustness. Following this principle, we incorporate L_1-norm into (3) to model a predictor with an L_1-and-L_2-norm-oriented *Loss* on inner product space as follows:

$$\begin{aligned}&\arg\min\varepsilon\left(X^I,Y^I\right)\\&=\underbrace{\frac{1}{2}\alpha_1^I\sum_{m_{u,s}\in M_K}\left|m_{u,s}-\sum_{k=1}^{d}x_{u,k}^I y_{k,s}^I\right|}_{L_1-\text{norm}-\text{oriented}Loss}+\underbrace{\frac{1}{2}\alpha_2^I\sum_{m_{u,s}\in M_K}\left(m_{u,s}-\sum_{k=1}^{d}x_{u,k}^I y_{k,s}^I\right)^2}_{L_2-\text{norm}-\text{oriented}Loss}\\&+\underbrace{\frac{1}{2}\lambda\sum_{m_{u,s}\in M_K}\left(\sum_{k=1}^{d}\left(x_{u,k}^I\right)^2+\sum_{k=1}^{d}\left(y_{k,s}^I\right)^2\right)}_{\text{Regularization}}\end{aligned}\tag{5}$$

where $\boldsymbol{\alpha}_1^I$ and $\boldsymbol{\alpha}_2^I$ denotes the coefficients of L_1-norm- and L_2-norm-oriented *Losses* on inner product space, respectively, $\boldsymbol{\alpha}_{u,k}^I$ denotes the u-th row and k-th column entry of X^I, $y_{k,s}^I$ denotes the k-th row and s-th column entry of Y^I, and X^I and Y^I denote the X and Y on inner product space. In (5), it is necessary to ensure that $\boldsymbol{\alpha}_1^I+\boldsymbol{\alpha}_2^I=1$ and $\boldsymbol{\alpha}_1^I+\boldsymbol{\alpha}_2^I\geq 0$. According to [21], besides, the bias is helpful for improving an LFA model's representation learning ability. Hence, we incorporate the bias into (5) as follows:

$$\begin{aligned}&\arg\min\varepsilon\left(X^I,Y^I\right)\\&=\underbrace{\frac{1}{2}\alpha_1^I\sum_{m_{u,s}\in M_K}\left|m_{u,s}-\sum_{k=1}^{d}x_{u,k}^I y_{k,s}^I-b_u^I-b_s^I-u\right|}_{L_1\text{-norm-oriented}Loss}\\&+\underbrace{\frac{1}{2}\alpha_2^I\sum_{m_{u,s}\in M_K}\left(m_{u,s}-\sum_{k=1}^{d}x_{u,k}^I y_{k,s}^I-b_u^I-b_s^I-u\right)^2}_{L_2\text{-norm-oriented}Loss}\\&+\underbrace{\frac{1}{2}\lambda\sum_{m_{u,s}\in M_K}\left(\sum_{k=1}^{d}\left(x_{u,k}^I\right)^2+\sum_{k=1}^{d}\left(y_{k,s}^I\right)^2+\left(b_u^I\right)^2+\left(b_s^I\right)^2\right)}_{\text{Regularization}},\end{aligned}\tag{6}$$

where u denotes the global average of MK, b_u^I and b_s^I are two scalars indicating the bias of u-th user and s-th service, respectively. Following (4), (6) can be minimized by SGD to train the desired X^I and Y^I. Notably, since the L_1-norm-oriented *Loss* term is not differentiable, it is required to be extended to the form on a single invocation $m_{u,s}$ as follows:

$$\begin{aligned}&\text{On}m_{u,s}:\\&\begin{cases}\Delta_{u,s}^I\geq 0:\varepsilon_{u,s}\left(X^I,Y^I\right)\\=\frac{1}{2}\left[\alpha_1^I\Delta_{u,s}^I+\alpha_2\left(\Delta_{u,s}^I\right)^2+\lambda\left(\sum_{k=1}^{d}\left(\left(x_{u,k}^I\right)^2+\left(y_{k,s}^I\right)^2\right)+\left(b_u^I\right)^2+\left(b_s^I\right)^2\right)\right]\\\Delta_{u,s}^I<0:\varepsilon_{u,s}\left(X^I,Y^I\right)\\=\frac{1}{2}\left[-\alpha_2^I\Delta_{u,s}^I+\alpha_2\left(\Delta_{u,s}^I\right)^2+\lambda\left(\sum_{k=1}^{d}\left(\left(x_{u,k}^I\right)^2+\left(y_{k,s}^I\right)^2\right)+\left(b_u^I\right)^2+\left(b_s^I\right)^2\right)\right]\end{cases}\end{aligned}\tag{7}$$

where the error $\Delta_{u,s}^I=m_{u,s}$-$\sum_{k=1}^{d}\left(x_{u,k}^I\,y_{k,s}^I\right)$-$b_u^I$-$b_s^I$-$u$. Then, by employing SGD to minimize (7), we have the training rules for the desired X^I and Y^I as follows:

$$\begin{aligned}&\text{On}m_{u,s},\forall k\in\{1,2,\ldots,d\}:\\&\begin{cases}\Delta_{u,s}^I\geq 0:\begin{cases}x_{u,k}^I\leftarrow x_{u,k}^I+\frac{1}{2}\alpha_1^I\eta y_{k,s}^I+\alpha_2^I\eta y_{k,s}^I\Delta_{u,s}^I-\eta\lambda x_{u,k}^I\\y_{k,s}^I\leftarrow y_{k,s}^I+\frac{1}{2}\alpha_1^I\eta x_{u,k}^I+\alpha_2^I\eta x_{u,k}^I\Delta_{u,s}^I-\eta\lambda y_{k,s}^I\\b_u^I\leftarrow b_u^I+\frac{1}{2}\alpha_1^I\eta+\alpha_2^I\eta\Delta_{u,s}^I-\eta\lambda b_u^I\\b_s^I\leftarrow b_s^I+\frac{1}{2}\alpha_1^I\eta+\alpha_2^I\eta\Delta_{u,s}^I-\eta\lambda b_s^I\end{cases}\\\Delta_{u,s}^I<0:\begin{cases}x_{u,k}^I\leftarrow x_{u,k}^I-\frac{1}{2}\alpha_1^I\eta y_{k,s}^I+\alpha_2^I\eta y_{k,s}^I\Delta_{u,s}^I-\eta\lambda x_{u,k}^I\\y_{k,s}^I\leftarrow y_{k,s}^I-\frac{1}{2}\alpha_1^I\eta x_{u,k}^I+\alpha_2^I\eta x_{u,k}^I\Delta_{u,s}^I-\eta\lambda y_{k,s}^I\\b_u^I\leftarrow b_u^I-\frac{1}{2}\alpha_1^I\eta+\alpha_2^I\eta\Delta_{u,s}^I-\eta\lambda b_u^I\\b_s^I\leftarrow b_s^I-\frac{1}{2}\alpha_1^I\eta+\alpha_2^I\eta\Delta_{u,s}^I-\eta\lambda b_s^I\end{cases}\end{cases}\end{aligned}\tag{8}$$

In (8), we adaptively adjust $\boldsymbol{\alpha}_1^I$ and $\boldsymbol{\alpha}_2^I$ following [17], i.e., $\boldsymbol{\alpha}_1^I$ and $\boldsymbol{\alpha}_2^I$ are self-adaptive according to the training errors of L_1-norm- and L_2-norm-oriented *Losses*. First, we define the following related definitions.

Definition 3. *Let* $\mathrm{L}_1^{\mathrm{I}}(\mathrm{n})$ *and* $\mathrm{L}_2^{\mathrm{I}}(\mathrm{n})$ *represent the partial* Loss *of* L_1*-norm- and* L_2*-norm-oriented* Losses *at the* n*-th training iteration on inner produce space, respectively. Then,* $\mathrm{L}_1^{\mathrm{I}}(\mathrm{n})$ *and* $\mathrm{L}_2^{\mathrm{I}}(\mathrm{n})$ *can be defined as follows:*

$$L_1^I(n)=\sum_{m_{u,s}\in M_K}\left|\Delta_{u,s}^I(n)\right|,L_2^I(n)=\sum_{m_{u,s}\in M_K}\left(\Delta_{u,s}^I(n)\right)^2\tag{9}$$

where $\Delta_{\mathrm{u,s}}^{\mathrm{I}}(\mathrm{n})$ *denotes the state of* $\Delta_{\mathrm{u,s}}^{\mathrm{I}}$ *at* n*-th training iteration.*

Definition 4. *Let* $\mathrm{C}_{\mathrm{L1}}^{\mathrm{I}}(\mathrm{n})$ *and* $\mathrm{C}_{\mathrm{L2}}^{\mathrm{I}}(\mathrm{n})$ *represent the cumulative* Loss *of* $\mathrm{C}_1^{\mathrm{I}}(\mathrm{n})$ *and* $\mathrm{C}_2^{\mathrm{I}}(\mathrm{n})$ *at the* n*-th training iteration on inner produce space, respectively. Then* $\mathrm{C}_{\mathrm{L1}}^{\mathrm{I}}(\mathrm{n})$ *and* $\mathrm{C}_{\mathrm{L2}}^{\mathrm{I}}(\mathrm{n})$ *can be computed by:*

$$C_{L_1}^I(n)=\sum_{j=1}^{n}L_1^I(j),C_{L_2}^I(n)=\sum_{j=1}^{n}L_2^I(j).\tag{10}$$

According to the cumulative *Loss* in the training process, $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ can be adaptively adjusted as follows:

$$\begin{aligned}\alpha_1^I(n)&=\frac{e^{-\gamma^I(n)C_{L_1}^I(n-1)}}{e^{-\gamma^I(n)C_{L_1}^I(n-1)}+e^{-\gamma^I(n)C_{L_2}^I(n-1)}},\\\alpha_2^I(n)&=\frac{e^{-\gamma^I(n)C_{L_2}^I(n-1)}}{e^{-\gamma^I(n)C_{L_1}^I(n-1)}+e^{-\gamma^I(n)C_{L_2}^I(n-1)}},\end{aligned}\tag{11}$$

where $\boldsymbol{\alpha}_1^I(n)$ and $\boldsymbol{\alpha}_2^I(n)$ denotes the state of $\boldsymbol{\alpha}_1^I$ and $\boldsymbol{\alpha}_2^I$ at n-th training iteration, respectively, and $\boldsymbol{\gamma}^I(n)$ is a balance coefficient that can be set adaptively at each training iteration as follows:

$$\gamma^I(n) = \frac{1}{C_{L_1}^I(n-1) + C_{L_2}^I(n-1)} \tag{12}$$

Therefore, based on (8), (9), (10), (11), and (12), we can train a QoS predictor with an L_1-and-L_2-norm-oriented *Loss* on inner product space, named D^{2}E-LF-1.

3.2 Predictor on Euclidean Distance Space (D^{2}E-LF-2)

As discussed in Example 1, the distance space can capture the local similarity of users or services. Hence, we model a predictor on the distance space. Unlike D^{2}E-LF-1 modeled on inner produce space, the predictions are achieved by computing the distance of two latent factor matrices. Similar to (5), we adopt an L_1-and-L_2-norm-oriented *Loss* to model a predictor on Euclidean distance space. Besides, the bias is also considered. Then, the modeled predictor is as follows:

$$\begin{aligned}
&\arg\min \varepsilon\left(X^D, Y^D\right) \\
&= \underbrace{\frac{1}{2}\alpha_1^D \sum_{m_{u,s}\in M_K} \left| m_{u,s} - \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - b_u^D - b_s^D - u \right|}_{L_1-\text{norm}-\text{oriented}Loss} \\
&\underbrace{+\frac{1}{2}\alpha_2^D \sum_{m_{u,s}\in M_K} \left(m_{u,s} - \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - b_u^D - b_s^D - u \right)^2}_{L_2-\text{norm}-\text{oriented}Loss} \\
&\underbrace{+\frac{1}{2}\lambda \sum_{m_{u,s}\in M_K} \left(\sum_{k=1}^{d}\left(x_{u,k}^D\right)^2 + \sum_{k=1}^{d}\left(y_{k,s}^D\right)^2 + \left(b_u^D\right)^2 + \left(b_s^D\right)^2 \right)}_{\text{Regularization}}
\end{aligned} \tag{13}$$

where $\boldsymbol{\alpha}_1^D$ and $\boldsymbol{\alpha}_2^D$ denote the coefficients of L_1-norm- and L_2-norm-oriented *Losses* on distance space, respectively, $\boldsymbol{\alpha}_{u,k}^D$ denotes the u-th row and k-th column entry of X^D, $y_{k,s}^D$ denotes the k-th row and s-th column entry of Y^D, X^D and Y^D denote the X and Y on distance space, and b_u^D and b_s^D are scalars indicating the bias of u-th user and s-th service on distance space, respectively. Following (7), (13) is required to be extended to the form on a single invocation $m_{u,s}$ as follows:

$$\begin{aligned}
&\text{On} m_{u,s}: \\
&\begin{cases}
\Delta_{u,s}^D \geq 0 : \varepsilon_{u,s}\left(X^D, Y^D\right) \\
= \frac{1}{2}\left[\alpha_1^I \Delta_{u,s}^D + \alpha_2 \left(\Delta_{u,s}^D\right)^2 + \lambda\left(\sum_{k=1}^{d}\left(\left(x_{u,k}^D\right)^2 + \left(y_{k,s}^D\right)^2\right) + \left(b_u^D\right)^2 + \left(b_s^D\right)^2\right)\right] \\
\Delta_{u,s}^D < 0 : \varepsilon_{u,s}\left(X^D, Y^D\right) \\
= \frac{1}{2}\left[-\alpha_1^I \Delta_{u,s}^D + \alpha_2 \left(\Delta_{u,s}^D\right)^2 + \lambda\left(\sum_{k=1}^{d}\left(\left(x_{u,k}^D\right)^2 + \left(y_{k,s}^D\right)^2\right) + \left(b_u^D\right)^2 + \left(b_s^D\right)^2\right)\right]
\end{cases},
\end{aligned} \tag{14}$$

where $\Delta_{u,s}^D = m_{u,s} - \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - b_u^D - b_s^D - u$. Then, by employing SGD to minimize (14), we have the training rules for the desired X^D and Y^D as follows:

$$\begin{aligned}
&\text{On} m_{u,s}, \forall k \in \{1, 2, \ldots, d\}: \\
&\begin{cases}
\Delta_{u,s}^D \geq 0 : \begin{cases}
x_{u,k}^D \leftarrow x_{u,k}^D + \frac{1}{2}\alpha_1\eta\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} \\
+\alpha_2\eta\Delta_{u,s}^D\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - \eta\lambda x_{u,k}^D \\
y_{s,k}^D \leftarrow y_{s,k}^D - \frac{1}{2}\alpha_1\eta\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} \\
-\alpha_2\eta\Delta_{u,s}^D\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - \eta\lambda y_{s,k}^D \\
b_u^D \leftarrow b_u^D + \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta_{u,s}^D - \eta\lambda b_u^D \\
b_s^D \leftarrow b_s^D + \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta_{u,s}^D - \eta\lambda b_s^D
\end{cases} \\
\Delta_{u,s}^D < 0 : \begin{cases}
x_{u,k}^D \leftarrow x_{u,k}^D - \frac{1}{2}\alpha_1\eta\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} \\
+\alpha_2\eta\Delta_{u,s}^D\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - \eta\lambda x_{u,k}^D \\
y_{s,k}^D \leftarrow y_{s,k}^D + \frac{1}{2}\alpha_1\eta\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} \\
-\alpha_2\eta\Delta_{u,s}^D\left(x_{u,k}^D - y_{s,k}^D\right) \Big/ \sqrt{\sum_{k=1}^{d}\left(x_{u,k}^D - y_{k,s}^D\right)^2} - \eta\lambda y_{s,k}^D \\
b_u^D \leftarrow b_u^D - \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta_{u,s}^D - \eta\lambda b_u^D \\
b_s^D \leftarrow b_s^D - \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta_{u,s}^D - \eta\lambda b_s^D
\end{cases}
\end{cases}.
\end{aligned} \tag{15}$$

Similar to (11), $\boldsymbol{\alpha}_1^D$ and $\boldsymbol{\alpha}_2^D$ can be self-adaptive according to the training errors of L_1-norm- and L_2-norm-oriented *Losses*. For the soundness of readability, we give the formulas of setting $\boldsymbol{\alpha}_1^D$ and $\boldsymbol{\alpha}_2^D$ directly as follows:

$$L_1^D(n) = \sum_{m_{u,s}\in M_K}\left|\Delta_{u,s}^D(n)\right|, L_2^D(n) = \sum_{m_{u,s}\in M_K}\left(\Delta_{u,s}^D(n)\right)^2, \tag{16}$$

$$C_{L_1}^D(n) = \sum_{j=1}^{n} L_1^D(j), C_{L_2}^D(n) = \sum_{j=1}^{n} L_2^D(j), \tag{17}$$

$$\begin{aligned}
\alpha_1^D(n) &= \frac{e^{-\gamma^D(n)C_{L_1}^D(n-1)}}{e^{-\gamma^D(n)C_{L_1}^D(n-1)} + e^{-\gamma^D(n)C_{L_2}^D(n-1)}}, \\
\alpha_2^D(n) &= \frac{e^{-\gamma^D(n)C_{L_2}^D(n-1)}}{e^{-\gamma^D(n)C_{L_1}^D(n-1)} + e^{-\gamma^D(n)C_{L_2}^D(n-1)}},
\end{aligned} \tag{18}$$

$$\gamma^D = \frac{1}{C_{L_1}^D(n-1) + C_{L_2}^D(n-1)}, \tag{19}$$

where $L_1^D(n)$ and $L_2^D(n)$ denote the partial *Loss* of L_1-norm- and L_2-norm-oriented *Losses*, respectively, $\Delta_{u,s}^D(n)$ denotes the state of $\Delta_{u,s}^D$, $C_{L1}^D(n)$ and $C_{L2}^D(n)$ denotes the cumulative *Loss* of $L_1^D(n)$ and $L_2^D(n)$, respectively, $\boldsymbol{\alpha}_1^D(n)$ and $\boldsymbol{\alpha}_2^D(n)$ denotes the state of $\boldsymbol{\alpha}_1^D$ and $\boldsymbol{\alpha}_2^D$, respectively, and $\boldsymbol{\gamma}^D(n)$ is a balance coefficient; all of them are defined at n-th training iteration on distance space.

Finally, based on (15), (16), (17), (18), and (19), we can train a QoS predictor with an L_1-and-L_2-norm-oriented *Loss* on distance space, named D^{2}E-LF -2.

3.3. Ensemble of D^{2}E-LF-1 and D^{2}E-LF-2

In machine learning, we always try to build a good enough model while the actual situation is often not ideal. However, we may get several not-so-good models with their respective merits. Under such a situation, the ensemble is a great way to integrate their merits [11], [22]. An effective ensemble requires that the base models are diversified and highly accurate [22],

TABLE 1
Algorithm D^{2}E-LF

Steps	Input: M_K
1	**initialize** f, λ, η, $\boldsymbol{\alpha}I\ 1 = \boldsymbol{\alpha}I\ 2 = \boldsymbol{\alpha}D\ 1 = \boldsymbol{\alpha}D\ 2 = w = 0.5$, $N = 1000$
2	**initialize** X^I, Y^I, X^D, and Y^D randomly
3	**for** $u = 1$ to $\lvert U \rvert$
4	**initialize** $bI\ u$ and $bD\ u$ randomly
5	**end for**
6	**for** $s = 1$ to $\lvert S \rvert$
7	**initialize** $bI\ s$ and $bD\ s$ randomly
8	**end for**
9	**while** $n \leq N$ && not converge
10	**for** each rating $m_{u,s}$ in M_K
11	**for** $k = 1$ to d
12	**update** $xI\ u,k$, $yI\ k,s$, $bI\ u$, and $bI\ s$ according to (8)
13	**update** $xD\ u,k$, $yD\ k,s$, $bD\ u$, and $bD\ s$ according to (15)
14	**end for**
15	**end for**
16	**update** $\boldsymbol{\alpha}I\ 1(n)$ and $\boldsymbol{\alpha}I\ 2(n)$ according to (9)–(12)
17	**update** $\boldsymbol{\alpha}D\ 1(n)$ and $\boldsymbol{\alpha}D\ 2(n)$ according to (16)–(19)
18	**update** w according to (21)–(24)
19	**for** each rating $m_{u,s}$ in M_K
20	**predict** $m_{u,s}$ to obtain $\hat{m}_{u,s}$ according to (20)
21	**end for**
22	$n = n + 1$
23	**end while**

[23]. In this study, D^{2}E-LF-1 and D^{2}E-LF-2 are modeled on inner produce space and distance space, respectively. Besides, an LFA-based model has been demonstrated to perform well in QoS prediction [8], [9]. Hence, D^{2}E-LF-1 and D^{2}E-LF-2 satisfy the two requirements of the ensemble. To ensemble D^{2}E-LF-1 and D^{2}E-LF-2, an effortless way is to adopt the weighting strategy to ensemble their predictions to obtain the final prediction $\hat{m}_{u,s}$ as follows:

$$\hat{m}_{u,s} = w\left(\sum_{k=1}^{d} x^I_{u,k} y^I_{k,s} + b^I_u + b^I_s + u\right) + (1-w)\left(\sqrt{\sum_{k=1}^{d}\left(x^D_{u,k} - y^D_{k,s}\right)^2} + b^D_u + b^D_s + u\right) \quad (20)$$

where w and (1-w) are the weights of the predictions of D^{2}E-LF-1 and D^{2}E-LF-2, respectively. Similar to $\boldsymbol{\alpha}I$ 1 and $\boldsymbol{\alpha}I$ 2 or $\boldsymbol{\alpha}D$ 1 and $\boldsymbol{\alpha}D$ 2, we can also make w adaptive according to the prediction errors of D^{2}E-LF-1 and D^{2}E-LF-2. To this end, we first measure the prediction errors $E^I(n)$ and $E^D(n)$ of D^{2}E-LF-1 and D^{2}E-LF-2 at the n-th training iteration, respectively, as follows:

$$E^I(n) = \sum_{m_{u,s} \in M_K} \left|\Delta^I_{u,s}(n)\right|, E^D(n) = \sum_{m_{u,s} \in M_K} \left|\Delta^D_{u,s}(n)\right|. \quad (21)$$

Then, we measure the sum errors $SI\ E(n)$ and $SD\ E(n)$ of $E^I\ (n)$ and $E^D(n)$ until the n-th training iteration, respectively, as follows:

$$S^I_E(n) = \sum_{j=1}^{n} E^I(j), S^D_E(n) = \sum_{j=1}^{n} E^D(j). \quad (22)$$

Finally, according to the sum errors $SI\ E(n)$ and $SD\ E(n)$, w can be adaptively adjusted during the training processes as follows:

$$w(n) = \frac{e^{-\gamma^E(n) S^I_E(n-1)}}{e^{-\gamma^E(n) S^I_E(n-1)} + e^{-\gamma^E(n) S^D_E(n-1)}} \quad (23)$$

where $w(n)$ denotes w at n-th training iteration and $\boldsymbol{\gamma}^E(n)$ is a balance coefficient to control the ensemble of D^{2}E-LF-1 and D^{2}E-LF-2 during the training processes. $\boldsymbol{\gamma}^E(n)$ can also be set adaptively as follows:

$$\gamma^E(n) = \frac{1}{S^I_E(n-1) + S^D_E(n-1)} \quad (24)$$

3.4 Algorithm Design and Analysis

Based on the above analyses, we design the algorithm for D^{2}E-LF. Table 1 represents the pseudo-code of Algorithm D^{2}E-LF. Notably, due to $(\lvert U \rvert + \lvert S \rvert) \ll \lvert M_K \rvert$ in real-world applications, the most time cost of Algorithm D^{2}E-LF is to train X^I, Y^I, X^D, and Y^D. Then, we deduce that Algorithm D^{2}E-LF's time complexity is $O(N \times \lvert M_K \rvert \times d)$. In addition, considering space complexity, Algorithm D^{2}E-LF adopts the following data structures: 1) two arrays with length $\lvert M_K \rvert$ to cache known data M_K and corresponding predictions, 2) two matrices with size $\lvert U \rvert \times d$ to cache X^I and X^D, 3) two matrices with size $\lvert S \rvert \times d$ to cache Y^I and Y^D, 4) two arrays with length $\lvert U \rvert$ to cache $bI\ u$ and $bD\ u$for each user, and 5) two arrays with length $\lvert S \rvert$ to cache $bI\ s$ and $bD\ s$ for each service. Then, Algorithm D^{2}E-LF's space complexity is $\Theta(d \times \max\{\lvert M_K \rvert / d, \lvert U \rvert, \lvert S \rvert\})$. Therefore, a D^{2}E-LF model's time and space complexities are both linear with $\lvert M_K \rvert$, which is easy to resolve in real applications.

4 Experiments

In the subsequent experiments, we aim at answering the following research questions (RQs):

- RQ. 1. Does the proposed D2E-LF model outperform state-of-the-art QoS predictors?
- RQ. 2. How do the inner product space and distance space influence D2E-LF's prediction accuracy?
- RQ. 3. What are the differences between training on inner produce space and distance space?

4.1 General Settings

Datasets. We choose two benchmarks QoS datasets of *Response Time* and *Throughput* in the experiments. They can be downloaded from a real system of WS-DREAM[1] and are frequently adopted in prior studies [12], [13], [14], [15], [16]. Response Time and Throughput have 1873838 and 1831253 records, respectively, generated by 339 users on 5825 different services. The two datasets are divided into training and testing sets with different proportions. The detailed information of these divided datasets is shown in Table 2.

Evaluation Metrics. The task of QoS prediction is to predict the missing data of a sparse QoS matrix. Thus, we test the errors between the predictions and the ground truths on the testing set to evaluate the accuracy of a QoS predictor. The commonly used evaluation metrics are root mean

1. http://wsdream.github.io/

TABLE 2
The Properties of All the Divided Datasets

Dataset	No.	Density*	Training set	Testing set
Response Time	D1.1	1%	18738	1855100
	D1.2	5%	93692	1780146
	D1.3	10%	187384	1686454
	D1.4	15%	374768	1499070
	D1.5	20%	749535	1124303
Throughput	D2.1	1%	18313	1812940
	D2.2	5%	91563	1739690
	D2.3	10%	183125	1739690
	D2.4	15%	274689	1648128
	D2.5	20%	366251	1465002

*Density denotes the percentage of training data compared to all the known records.

TABLE 3
Descriptions of All the Involved Models

Model	Description
BLFA[21]	The basic LFA model proposed for the recommender system. It is also widely used for QoS prediction.
RSNMF[11]	It incorporates non-negative constraints into LFA. It has a regularized single element-dependent form and is designed for QoS prediction.
NIMF[24]	It considers neighborhood information of similar users in constructing the LFA model. It is designed for QoS prediction.
NAMF [13]	It is a geography-aware QoS predictor. Its core is a network-aware LFA model.
GeoMF[12]	It improves LFA-based QoS predictors by considering the geographical neighborhoods of QoS data.
LMF-PP[16]	It is a location-based QoS predictor. It conducts the LFA with additional geographical information.
DCALF[8]	It is a data-characteristic-aware LFA model for generating highly accurate QoS prediction proposed. It considers the neighborhood information and outliers simultaneously.
LBFM [4]	It is a location-based factorization machine by making full use of the location, network, and country information of users and services.
AutoRec[26]	It is a deep-learning-based model [26]. It is an autoencoder framework for collaborative filtering. It is also widely used for QoS prediction.
D^{2}E-LF	The proposed double-space and double-norm LFA model in this paper.

square error (RMSE) and mean absolute error (MAE) [8], [9]. The smaller RMSE/MAE denotes the higher prediction accuracy. RMSE and MAE can be computed as follows:

$$RMSE = \sqrt{\left(\sum_{m_{u,s}\in\Gamma}\left(m_{u,s}-\hat{m}_{u,s}\right)^2\right)\Big/|\Gamma|},$$

$$MAE = \left(\sum_{m_{u,s}\in\Gamma}\left|m_{u,s}-\hat{m}_{u,s}\right|\right)\Big/|\Gamma|$$

where Γ denotes the testing set.

Baselines. We compare D^{2}E-LF with nine related state-of-the-art predictors. Table 3 gives brief descriptions of all the involved models. Most of them are LFA-based predictors. BLF and RSNMF are basic LFA-based models. NIMF, NAMF, GeoMF, LMF-PP, DCALF, and LBFM improve basic LFA-based models by considering neighborhood information. Besides, DCALF also considers the outliers issue. AutoRec is a deep-learning-based model. Notably, all of them are implemented on inner produce space.

Implementation Details. We set $d = 20$ for all predictors except for AutoRec as it is a deep-learning-based model to draw a fair comparison. We respectively tune the learning rate and regularization coefficient for all the predictors to achieve their own best prediction accuracy. Besides, the other hyperparameters of all the comparison predictors are set according to their original papers. Each dataset is tested five times, and we report the average results. The training termination conditions are set as: a) the error difference between two consecutive iterations becomes smaller than 10-6, or b) the number of consumed iterations reaches a preset threshold, i.e., 1000. The experimental environment is a personal computer (PC) with 16 cores i7 CPU, and 32 G RAM. To promote reproducible research, our code is available on: *https://github.com/Wuziqiao/D2E-LF.git*.

4.2 Performance Comparison (RQ. 1)

4.2.1 Comparison of Prediction Accuracy

Table 4 records the comparison results. To better analyze these results, we make statistical analyses of win/loss counts of D^{2}E-LF versus comparison models and the Friedman test [27]. The statistical results are summarized at the second-to-last and the last rows of Table 4, respectively. The win/loss counts, such as 20/0 in the column NAMF, indicate that D^{2}E-LF achieves better RMSE/MAE than BLFA on 20 cases and no one worse. Friedman rank test is a standard statistical method to verify the effect of multiple models on multiple datasets. The smaller value of F-rank indicates the lower RMSE/MAE. From Table 4, we have two main observations. First, D^{2}E-LF achieves lower RMSE/MAE than each comparison model on most cases. Concretely, the win/loss situation of D^{2}E-LF versus BLFA, RSNMF, NIMF, NAMF, GeoMF, LMF-PP, DCALF, LBFM, and AutoRec are 19/1, 18/2, 15/5, 20/0, 20/0, 17/3, 16/4, 12/8, and 19/1, respectively. The total win/loss situation is 156/24. Second, D^{2}E-LF achieves the lowest F-rank value of 2.2 among all the models, which indicates that D^{2}E-LF has a lower RMSE/MAE than the other comparison models on all the testing cases. Therefore, the two observations show that D^{2}E-LF has the highest prediction accuracy among all the models.

In addition, to evaluate whether D^{2}E-LF has significantly higher prediction accuracy than its peers, we also conduct the Wilcoxon signed-ranks test [27] on Table 4. It is a non-parametric pairwise comparison method and includes three indicators—$R+$, R- and p-value. The larger $R+$ denotes the better performance, and the p-value is the level of significance. Notably, we normalize the results of D1 and D2 of Table 4, respectively, because they have range differences. D^{2}E-LF is compared with each comparison model one by one. Table 5 records the results, where we observe that: 1) D^{2}E-LF achieves a much larger $R+$ than each comparison model, 2) all the p-values are smaller than 0.05 except for one case of D^{2}E-LF vs.LBFM, and 3) the p-value of D^{2}E-LF vs.LBFM is 0.0753, although it is not smaller than 0.05, it is still smaller than 0.1. Therefore, these three observations show that D^{2}E-LF has significantly higher prediction accuracy than its peers with a significance level of 0.1.

TABLE 4
The Comparison Results on Prediction Accuracy, Including Win/Loss Counts and Friedman Test, Where • Denotes That D^{2}E-LF Has a Lower RMSE/MAE Than Comparison Models

Datasets	Metric	BLFA	RSNMF	NIMF	NAMF	GeoMF	LMF-PP	DCALF	LBFM	AutoRec	D^{2}E-LF
D1.1	RMSE	1.7671•	1.7529•	1.7512•	1.7446•	1.5824•	1.6024•	1.6649•	1.6272•	1.7461•	1.5376
	MAE	0.7130•	0.6921•	0.6890•	0.6894 •	0.6781•	0.6632•	0.6530•	0.6124	0.7064•	0.6277
D1.2	RMSE	1.4058•	1.4032•	1.4023•	1.3995•	1.3152•	1.3410•	1.3731•	1.3468•	1.3730•	1.3056
	MAE	0.5561•	0.5438•	0.5502•	0.5465•	0.5305•	0.5285•	0.5127•	0.4558	0.5467•	0.4738
D1.3	RMSE	1.2657•	1.2689•	1.2698•	1.2694•	1.2190•	1.2419•	1.2450•	1.2741•	1.2678•	1.1994
	MAE	0.4944•	0.4868•	0.4842•	0.4976•	0.4827•	0.4725•	0.4544•	0.3792	0.5055•	0.4069
D1.4	RMSE	1.2155•	1.2067•	1.1906•	1.2178•	1.1742•	1.2102•	1.2001•	1.2463•	1.1923•	1.1493
	MAE	0.4691•	0.4492•	0.4508•	0.4625•	0.4495•	0.4472•	0.4346•	0.3614	0.4598•	0.3820
D1.5	RMSE	1.1885•	1.1568•	1.1649•	1.1592•	1.1528•	1.1612•	1.1759•	1.2014•	1.1681•	1.1256
	MAE	0.4531•	0.4371•	0.4346•	0.4360•	0.4366•	0.4260•	0.4246•	0.3478	0.4482•	0.3687
D2.1	RMSE	84.7661•	84.5423•	84.1428•	85.7821•	88.6521•	85.1048•	83.2891•	86.2188•	85.1214•	80.4369
	MAE	32.3966	32.9940	31.8251	34.8227•	37.2881•	31.9704	31.9826	33.4118•	32.5432	33.2675
D2.2	RMSE	51.6982•	60.7994•	51.4585•	53.9572•	57.7842•	51.7765•	51.4123•	54.6529•	55.5352•	50.9875
	MAE	18.9776•	21.4302•	17.7153	20.2104•	24.7465•	18.3091	18.62373•	19.6723•	21.3118•	18.5502
D2.3	RMSE	46.5606•	50.5298•	45.8432•	46.0215•	49.2456•	46.1418•	45.9013•	45.9237•	48.4771•	45.4089
	MAE	16.1924•	17.2305•	15.5264	17.0126•	22.4728•	15.9125	15.3430	15.8224	17.0310•	16.1166
D2.4	RMSE	43.6210•	45.2647•	43.0232•	43.6522•	45.3255•	42.9927•	42.6235•	42.4228•	44.5246•	41.9991
	MAE	14.9279•	14.6880	14.2146	15.6547•	17.7908•	14.7450•	14.0664	14.3665	15.0156•	14.7267
D2.5	RMSE	41.7846•	43.5882•	41.0765•	41.3523•	43.9845•	41.4084•	41.2194•	40.5023•	43.0654•	40.3075
	MAE	14.3061•	14.3654•	13.5638	14.6482•	16.2852•	14.1033•	13.5491	12.5482	14.2265•	14.0116
Statistic	Win/loss	19/1	18/2	15/5	20/0	20/0	17/3	16/4	12/8	19/1	**156/24***
	F-rank*	7.4	7.3	4.6	7.2	6.8	4.35	3.4	4.4	7.35	**2.2**

**The total win/loss cases of D^{2}E-LF.* * *A lower F-rank value indicates a higher prediction accuracy.*

4.2.2 Comparison of Computational Efficiency

Fig. 4 records the CPU running time of all the involved models, where we have two observations. First, AutoRec consumes the most CPU running time among all the models. The reason is that it is a deep-learning-based model that requires the complete data of a sparse QoS matrix to train [28], [29]. Second, D^{2}E-LF consumes a little more CPU running time than the other models. The reason is that D^{2}E-LF trains two predictors for ensemble. However, as analyzed in Section 3.4, D^{2}E-LF's time complexity is linear with the known data of a sparse QoS matrix. Therefore, D^{2}E-LF's computational efficiency is also suitable for real applications.

4.3 Influence of Inner Produce Space and Distance Space (RQ. 2)

In (20), the weight w is adaptive. To test the influence of inner produce and distance spaces, we fix the weight w during the training. In this set of experiments, the weight w is increased from 0 to 1. Note that when $w = 1$, D^{2}E-LF is trained on inner produce space only, i.e., D^{2}E-LF is equivalent to D^{2}E-LF-1.

TABLE 5
The Results of the Wilcoxon Signed-Ranks Test on Table 4

Comparison	$R+$	$R-$	p-value*
D^{2}E-LF vs. BLFA	204	6	**0.0001**
D^{2}E-LF vs. RSNMF	207	3	**0.0001**
D^{2}E-LF vs. NIMF	183	27	**0.0019**
D^{2}E-LF vs. NAMF	210	0	**0.0001**
D^{2}E-LF vs. GeoMF	210	0	**0.0001**
D^{2}E-LF vs. LMF-PP	194	16	**0.0009**
D^{2}E-LF vs. DCALF	185	25	**0.0015**
D^{2}E-LF vs.LBFM	144	66	**0.0753**
D^{2}E-LF vs. AutoRec	207	3	**0.0001**

**The accepted hypotheses with a significance level of 0.1 are highlighted.*

On the other hand, when $w = 0$, D^{2}E-LF is trained on distance space only, i.e., D^{2}E-LF is equivalent to D^{2}E-LF-2. Figs. 5 and 6 record the experimental results on D1 and D2, respectively, where a red circle marks the lowest RMSE/MAE. From them, we have the following significant findings:

- When testing RMSE on D1 and D2, all the cases show that the ensemble of inner produce and distance spaces achieves lower RMSE than D2E-LF-1 and D2E-

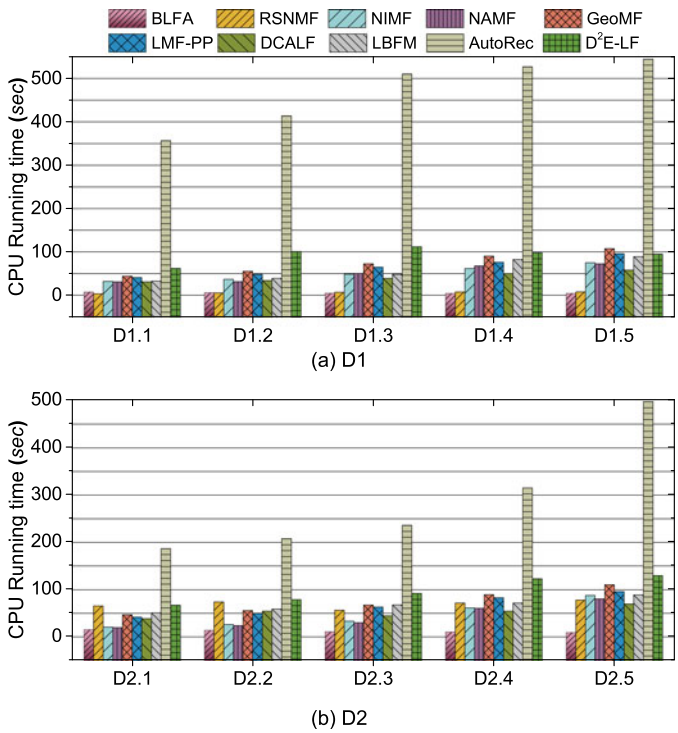


Fig. 4. CPU running time of comparison models.

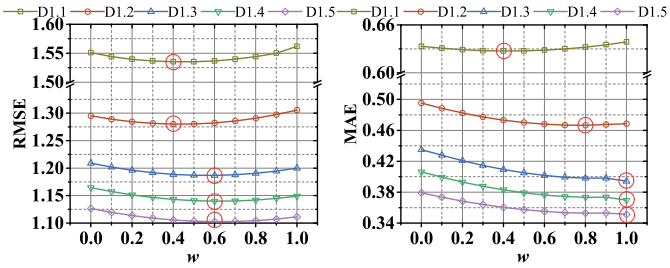


Fig. 5. The RMSE and MAE of D^{2}E-LF on D1 when w increasing from 0 to 1.

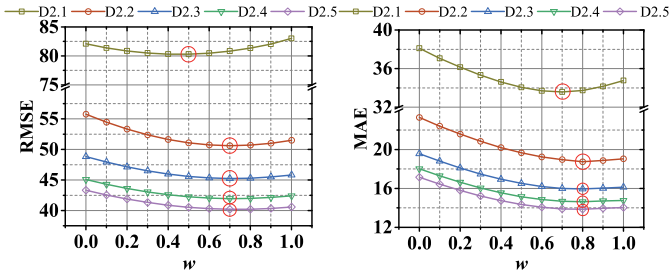


Fig. 6. The RMSE and MAE of D^{2}E-LF on D2 when w increasing from 0 to 1.

LF-2. For example, when w = 0.4 on D1.1, the lowest RMSE is 1.5353, which is 1.70% and 1.03% lower than that of D2E-LF-1 and D2E-LF-2, respectively.

- When testing MAE on D2, the results show that the ensemble of inner produce and distance spaces achieves lower MAE than D2E-LF-1 and D2E-LF-2. For example, when w = 0.8 on D2.2, the lowest MAE is 18.7435, which is 1.61% and 19.56% lower than that of D2E-LF-1 and D2E-LF-2, respectively.
- When testing MAE on D1, there are three cases where the ensemble of inner produce and distance spaces does not reduce the MAE of D2E-LF-1. The reason is that D2E-LF-2 has a much higher MAE than D2E-LF-1. Concretely, the MAEs of D2E-LF-2 are 0.4352, 0.4064, and 0.3793 on D1.3, D1.4, and D1.5, respectively, which are 10.35%, 9.94%, and 7.99 higher than that of D2E-LF-1. Then, D2E-LF-2 violates the ensemble principle that each base model requires to be accurate [22]. However, the adaptive strategy of (23) guarantees D2E-LF's performance under such a situation.

We conclude from the above findings that the ensemble of D^{2}E-LF-1 and D^{2}E-LF-2 is useful in improving D^{2}E-LF's accuracy, which verifies that both inner produce space and distance space can boost D^{2}E-LF's representation ability to a sparse QoS matrix. Although D^{2}E-LF is not significantly enhanced by the ensemble of inner produce space and distance space in a few cases, its prediction accuracy is still guaranteed.

4.4 The Differences Between Training on Inner Produce Space and Distance Space (RQ. 3)

The principle of D^{2}E-LF is to ensemble the two predictors of D^{2}E-LF-1 and D^{2}E-LF-2 that are trained on inner produce space and distance space, respectively. The core of the two predictors is their two latent factor matrices. Hence, to analyze the differences between training on inner produce space and distance space, we depict the distributions of latent factors of the two predictors, as shown in Fig. 7, 8, 9, and 10. To analyze these figures, we adopt a Gaussian function

$$f(v) = a + \frac{b}{\sigma\sqrt{\pi/2}} e^{-2\frac{(v-\mu)^2}{\sigma^2}}$$

to fit them, where v is the value of latent factor, $f(v)$ is the number of latent factors in each bin, a and b are the constants, $\boldsymbol{\mu}$ is the expectation, and $\boldsymbol{\sigma}$ is the standard deviation. Then, we have the following important findings:

- The distributions of latent factors on inner produce tend to be more centralized than on distance space. For example, the fitting parameters of inner space on D1.2 (Fig. 7b) are $\sigma = 0.1328$, full width at half maximum (FWHM) = 0.1563, and height = 3905.86, while that of distance space on D1.2 (Fig. 8b) are $\sigma = 0.7255$, FWHM = 0.8542, and height = 980.52.
- The distributions of latent factors on distance space tend to be symmetric w.r.t. zero while that on inner produce space does not. For example, the μ of distance space on D1.2 (Fig. 8b) is 0.0099, while that of inner produce space on D1.2 (Fig. 7b) is 0.0464.
- The values of latent factors on distance space are wider than that on inner produce space. For example, Figs.9 and 10 have different horizontal axis ranges. The maximum and minimum of latent factors on distance space on D2.2 (Fig. 10b) are 550.33 and -523.33, while that on inner produce space on D2.2 (Fig. 9b) are 28.37 and -18.62.

These findings obviously reveal that the distributions of latent factors trained on inner produce space and distance space are significantly different, demonstrating that the two predictors of D^{2}E-LF-1 and D^{2}E-LF-2 are diversified. According to the principle of ensemble [22], [23], such diversity is beneficial for boosting a base model. By ensembling D^{2}E-LF-1 and D^{2}E-LF-2, therefore, D^{2}E-LF's representation learning ability is enhanced.

4.5 Summary of Experiments

Based on the above experimental results, we summarize D^{2}E-LF's advantages and disadvantages as follows. Advantages: D^{2}E-LF integrates multi-merits originating from inner product space, distance space, L_1-norm, and L_2-norm, making it outperform nine state-of-the-art models in predicting the missing data of a sparse QoS matrix. Disadvantages: D^{2}E-LF consumes a little more CPU running time than eight state-of-the-art models.

5 Related Work

A QoS model is to evaluate the quality of Web services with the same or similar functionalities [1], [2], [3]. Generally, it is modeled based on various QoS properties [1]–[3], including response time, accessibility, throughput, availability, capacity, robustness, interoperability, authentication, cost, confidentiality, reputation, etc. However, the QoS data of these properties are not easy to collect by warm-up test because it is time-consuming and expensive [6], [7]. Alternatively, QoS

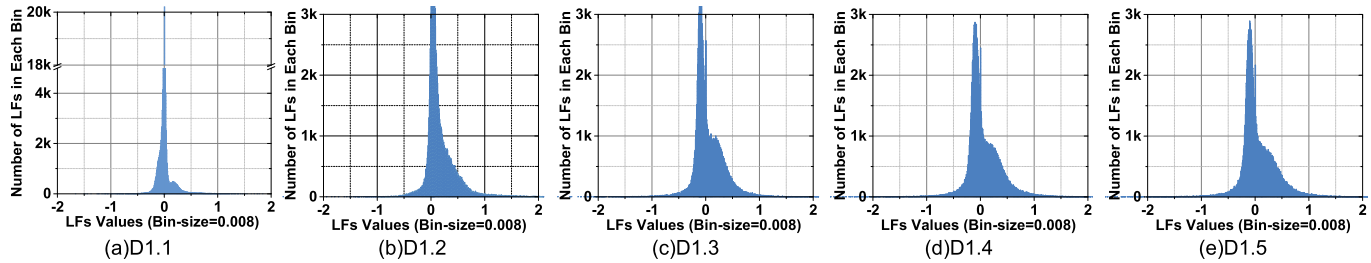


Fig. 7. The distributions histogram of latent factors (LFs) of D^{2}E-LF-1 (trained on inner produce space) on D1.

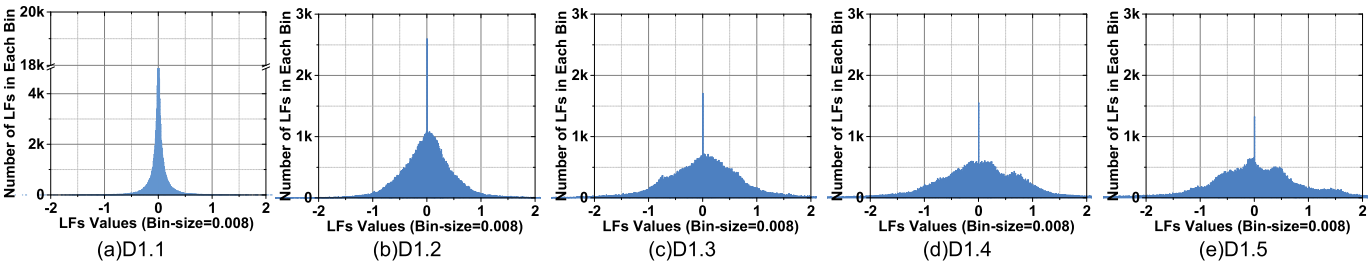


Fig. 8. The distributions histogram of latent factors (LFs) of D^{2}E-LF-2 (trained on distance space) on D1.

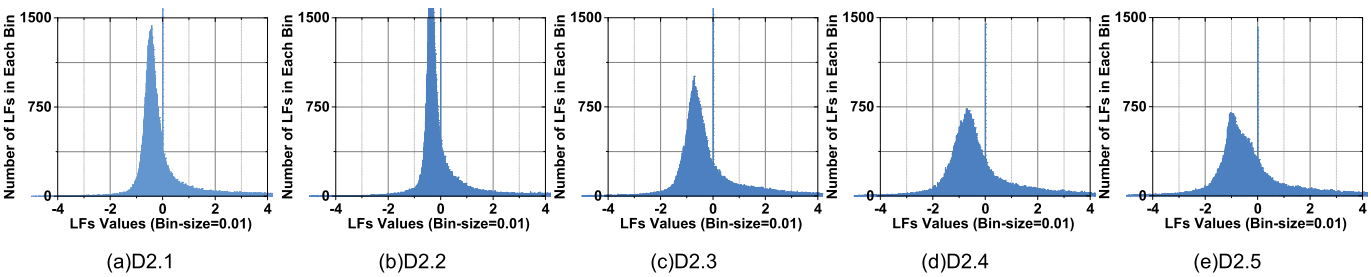


Fig. 9. The distributions histogram of latent factors (LFs) of D^{2}E-LF-1 (trained on inner produce space) on D2.

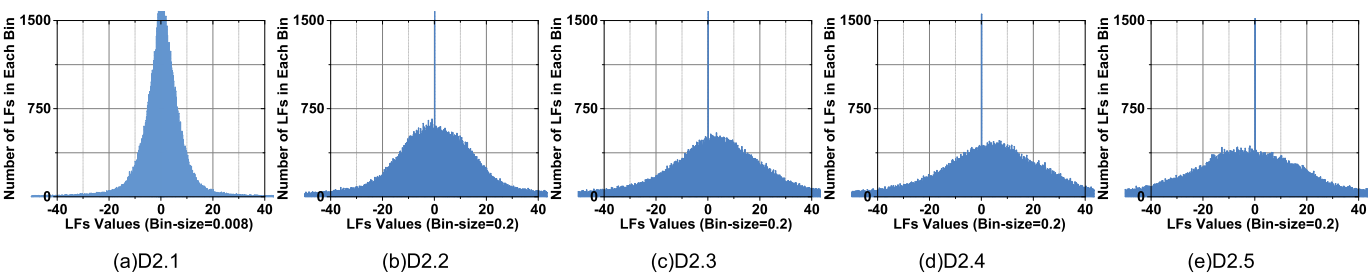


Fig. 10. The distributions histogram of latent factors (LFs) of D^{2}E-LF-2 (trained on distance space) on D2.

prediction is a practical way to obtain QoS data, which is the focus of this paper.

In a service-based application, a user's QoS data are mainly determined by his/her invoking environment [6], [8]. In other words, if two users' invoking environments are similar, their QoS data are probably similar. Hence, many QoS properties, e.g., response time and throughput, are user-dependent [4], [8], [50]. From this point of view, the most common approach of QoS prediction is collaborative filtering (CF) [30], [31], [32]. The early approach of CF is memory-based [7]. It considers one or several QoS properties as the metric to measure the similarity of users and/or services [6], [33], [34], [35]. As a result, it usually suffers from data sparsity problems because a user cannot touch all of the available services to acquire sufficient QoS data in real-world scenarios. On the other hand, model-based CF has made great progress in the last decade due to its effectiveness in dealing with data sparsity problems [6], [7]. Its key is to train a predefined prediction model based on the observed QoS data to predict the missing ones. Among numerous model-based CF approaches, a latent factor analysis (LFA)-based one is highly popular and widely investigated owing to its high efficiency and scalability [10], [11]. To date, various sophisticated LFA-based ones have been proposed, including non-negative constraint [11], asymmetric correlation regularized [36], data-characteristic-aware [8], neighborhood integrated [24], posterior-neighborhood-regularized [9], data transformation and adaptive weights based [37], generative probabilistic learning framework-based [38], and covering-based and neighborhood-aware [39] ones.

In addition, to further improve the accuracy of the LFA-based approach, many researchers consider the additional geographical information [12], [13], [14], [15], [16], [25] because users in the same area tend to possess similar invoking environments [13]. Their common point is to employ geographical information first to identify the neighborhoods of services and users. Then, the identified neighborhoods are incorporated into an LFA-based approach as a regularization constraint [8], [9]. For example, Zheng *et al.* considered the user network map to develop a network-aware LFA-based model [13]. Ryu *et al.* identified the neighborhoods based on the location of users and services [16]. Feng and Huang implemented a neighborhood-aware LFA-based model by systematically modeling sample set diversity, geographical information, and platform context [14]. Yang *et al.* proposed a location-based factorization machine by making full use of the location, network, and country information of users and services [4].

Recently, deep neural networks (DNNs) have attracted much attention in QoS prediction [40], [41], [42], [43], [44], [45], [46], [50]. The DNNs-based approaches exploit the powerful representation learning ability of DNNs [47], [48] to analyze the side information, such as context, to improve prediction accuracy. For example, Wu *et al.* proposed a universal deep neural model to make multiple attributes QoS prediction with contexts [40]. Gao *et al.* proposed a holistic framework for QoS prediction based on fuzzy clustering and neural collaborative filtering [42], where fuzzy clustering clusters contextual information and neural collaborative filtering leverages local and global features. Wang *et al.* proposed a hidden-state-aware network to generate explainable and fused features to help QoS prediction [46].

However, compared with the approaches mentioned above, the proposed D^{2}E-LF model possesses its significance in the following aspects:

- The LFA-based approaches are developed on inner product space with an L2-norm-oriented Loss only, which cannot comprehensively represent QoS data as inner product space and L2-norm have their respective limitations. In comparison, D2E-LF integrates multi-merits originating from inner product space, distance space, L1-norm, and L2-norm, making it achieve highly accurate QoS prediction.
- The DNNs-based approaches usually suffer from high computational costs inherited from DNNs [28], [29]. In comparison, D2E-LF has higher computational efficiency because its core component is a high-efficient LFA model [49] trained only on observed entries of a sparse QoS matrix.
- D2E-LF is also compatible with both geographical and side information. Such information can be incorporated into its objective functions (5) and (13) as the data-dependent regularization. We plan to investigate this point in the future.

6 Conclusion

This paper proposes a Double-space and Double-norm Ensembled Latent Factor (D^{2}E-LF) model for highly accurate Web service Quality-of-Service (QoS) prediction. Its main idea is three-fold: 1) Double-space—inner product space and distance space are employed to model two kinds of latent factor analysis-based QoS predictors, respectively, 2) Double-norm—both of these two predictors adopt an L_1-and-L_2-norm-oriented *Loss* function, and 3) Ensembled—building an ensemble of these two predictors by a weighting strategy. By doing so, D^{2}E-LF integrates multi-merits originating from inner product space, distance space, L_1-norm, and L_2-norm, making it achieve highly accurate QoS prediction. Extensive experiments on two real-world QoS datasets are conducted to evaluate D^{2}E-LF. The results demonstrate that it significantly outperforms its peers in terms of prediction accuracy. In the future, we will enhance D^{2}E-LF from two aspects. First, we plan to incorporate more metrics into its *Loss*, e.g., $L_{2,1}$-norm. Second, we plan to incorporate geographical and side information into its objective function as a regularization constraint.

References

[1] A. Ramírez, J. A. Parejo, J. R. Romero, S. Segura, and A. Ruiz-Cortés, "Evolutionary composition of QoS-aware web services: A many-objective perspective," *Expert Syst. Appl.*, vol. 72, pp. 357–370, 2017.

[2] O. Adeleye, J. Yu, G. Wang, and S. Yongchareon, "Constructing and evaluating evolving web-API Networks - A complex network perspective," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2021.3114709.

[3] A. S. D. Silva, H. Ma, Y. Mei, and M. Zhang, "A survey of evolutionary computation for web service composition: A technical perspective," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 4, pp. 538–554, Aug. 2020.

[4] Y. Yang, Z. Zheng, X. Niu, M. Tang, Y. Lu, and X. Liao, "A location-based factorization machine model for web service QoS prediction," *IEEE Trans. Serv. Comput.*, vol. 14, no. 5, pp. 1264–1277, Sep./Oct. 2021.

[5] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Computational intelligence based QoS-Aware web service composition: A systematic literature review," *IEEE Trans. Serv. Comput.*, vol. 10, no. 3, pp. 475–492, May/Jun. 2017.

[6] S. H. Ghafouri, S. M. Hashemi, and P. C. K. Hung, "A survey on web service QoS prediction methods," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2020.2980793.

[7] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service QoS prediction via collaborative filtering: A survey," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2020.2995571.

[8] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2525–2538, Jun. 2022, doi: 10.1109/TKDE.2020.3014302.

[9] D. Wu, Q. He, X. Luo, M. Shang, Y. He, and G. Wang, "A posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction," *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 793–805, Mar./Apr. 2022.

[10] A. Liu *et al.*, "Differential private collaborative Web services QoS prediction," *World Wide Web*, vol. 22, no. 6, pp. 2697–2720, 2019.

[11] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524–537, Mar. 2016.

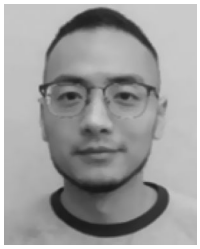
[12] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service QoS prediction," *Knowl.-Based Syst.*, vol. 138, pp. 188–201, 2017.

[13] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 126–137, Mar. 2016.

[14] Y. Feng and B. Huang, "Cloud manufacturing service QoS prediction based on neighbourhood enhanced matrix factorization," *J. Intell. Manuf.*, vol. 31, no. 7, pp. 1649–1660, 2020.

[15] S. S. Kumar and S. M. Anouncia, "QoS-based concurrent user-service grouping for web service recommendation," *Autom. Control Comput. Sci.*, vol. 52, no. 3, pp. 220–230, 2018.

[16] D. Ryu, K. Lee, and J. Baik, "Location-based web service QoS prediction via preference propagation to address cold start problem," *IEEE Trans. Serv. Comput.*, vol. 14, no. 3, pp. 736–746, May/Jun. 2021.
[17] D. Wu, M. Shang, X. Luo, and Z. Wang, "An L_1-and-L_2-Norm-oriented latent factor model for recommender systems," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3071392.
[18] S. Zhang, L. Yao, B. Wu, X. Xu, X. Zhang, and L. Zhu, "Unraveling metric vector spaces with factorization for recommendation," *IEEE Trans. Ind. Inform.*, vol. 16, no. 2, pp. 732–742, Feb. 2020.
[19] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 193–201.
[20] X. Zhu *et al.*, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation," *IEEE Trans. Serv. Comput.*, vol. 14, no. 3, pp. 889–902, May/Jun. 2021.
[21] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
[22] Z.-H. Zhou, "Ensemble learning," *Encyclopedia biometrics*, vol. 1, pp. 270–273, 2009.
[23] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 1–40, 2012.
[24] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, Third Quarter 2013.
[25] J. Li, H. Wu, J. Chen, Q. He, and C.-H. Hsu, "Topology-aware neural model for highly accurate QoS prediction," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1538–1552, Jul. 2022.
[26] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec:Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.
[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, no. pp. 1–30, 2006.
[28] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3553–3559.
[29] Z.-H. Zhou and J. Feng, "Deep forest," *Nat. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2019.
[30] H. Chen and J. Li, "Learning multiple similarities of users and items in recommender systems," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 811–816.
[31] H. Zhao, Q. Yao, J. T. Kwok, and D. L. Lee, "Collaborative filtering with social local models," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 645–654.
[32] D. M. Camacho, K. M. Collins, R. K. Powers, J. C. Costello, and J. J. Collins, "Next-generation machine learning for biological networks," *Cell*, vol. 173, no. 7, pp. 1581–1592, 2018.
[33] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1770–1782, Sep. 2018.
[34] C. Wang *et al.*, "Confidence-aware matrix factorization for recommender systems," in *Proc. 32nd AAAI Conf. Artif. Intell*, 2018, pp. 434–442.
[35] Y. He, C. Wang, and C. Jiang, "Correlated matrix factorization for recommendation with implicit feedback," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 3, pp. 451–464, Mar. 2019.
[36] Q. Xie, S. Zhao, Z. Zheng, J. Zhu, and M. R. Lyu, "Asymmetric correlation regularized matrix factorization for web service recommendation," in *Proc. IEEE Int. Conf. Web Serv.*, 2016, pp. 204–211.
[37] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online qos prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, Oct. 2017.
[38] Z. Luo, L. Liu, J. Yin, Y. Li, and Z. Wu, "Latent ability model: A generative probabilistic learning framework for workforce analytics," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 923–937, May 2019.
[39] Y. Zhang *et al.*, "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 14, no. 5, pp. 1333–1344, Sep./Oct. 2021.
[40] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. H. Hsu, "Multiple attributes QoS prediction via deep neural model with contexts," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 1084–1096, Jul./Aug. 2021.
[41] G. White, A. Palade, C. Cabrera, and S. Clarke, "Autoencoders for QoS prediction at the Edge," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2019, pp. 1–9.
[42] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, May 2020.
[43] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, and Z. Mai, "QoS prediction for service recommendation with features learning in mobile Edge computing environment," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1136–1145, Dec. 2020.
[44] P. Sahu, S. Raghavan, and K. Chandrasekaran, "Ensemble deep neural network based quality of service prediction for cloud service recommendation," *Neurocomputing*, vol. 465, pp. 476–489, 2021.
[45] G. Zou, J. Chen, Q. He, K. C. Li, B. Zhang, and Y. Gan, "NDMF: Neighborhood-integrated deep matrix factorization for service QoS prediction," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2717–2730, Dec. 2020.
[46] Z. Wang, X. Zhang, M. Yan, L. Xu, and D. Yang, "HSA-Net: Hidden-state-aware networks for high-precision QoS prediction," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1421–1435, Jun. 2022.
[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[48] A. Saxe, S. Nelli, and C. Summerfield, "If deep learning is the answer, what is the question?," *Nature Rev. Neurosci.*, vol. 22, no. 1, pp. 55–67, 2021.
[49] S. Rendle, W. Krichene, L. Zhang, and J. R. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 240–248.
[50] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-aware deep collaborative filtering for service recommendation," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 51, no no. 6, pp. 3796–3807, Jun. 2021.



Di Wu (Member, IEEE) received the PhD degree from the Chongqing Institute of Green and IntelligentnTechnology (CIGIT), Chinese Academy of Sciences (CAS), China, in 2019, and then joined CIGIT, CAS, China. He is currently a professor with the College of Computer and Information Science, Southwest University, Chongqing, China. He has more than 50 publications, including journals of *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Services Computing*, etc., and conferences of ICDM, WWW, IJCAI, ECAI, etc. His research interests include machine learning and data mining. His homepage: https://wuziqiao.github.io/Homepage/

Peng Zhang (Student Member, IEEE) received the BS degree in network engineering from the Shanxi Datong University, Datong, China, in 2018. He is currently working toward the MS degree in computer technology with the Chongqing University of Posts and Telecommunications, Chongqing, China. He is a visiting student from July 2020 to present at the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences (CAS), Chongqing, China. His current research interests include data mining, big data analysis, and artificial intelligence.

Yi He (Member, IEEE) received the PhD degree from the University of Louisiana at Lafayette, Lafayette, LA, in 2020, and then joined the Old Dominion University, USA where he is an Assistant Professor. He has more than 20 publications in top-tier venues–AAAI, IJCAI, WWW, ICDM, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, etc. His research interests include lie broadly in machine learning, data mining, and optimization theory.

Xin Luo (Senior Member, IEEE) received the BS degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the PhD degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a professor with the Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored more than 200 papers (including more than 80 IEEE Transactions papers) in the areas of his interests. His research interests include big data analysis and intelligent control. He was the recipient of the outstanding associate editor Award from IEEE/CAA Journal of Automatica Sinica in 2020. He is currently serving as an associate editor for *IEEE Transactions on Neural Networks and Learning Systems*, and *IEEE/CAA Journal of Automatica Sinica*.